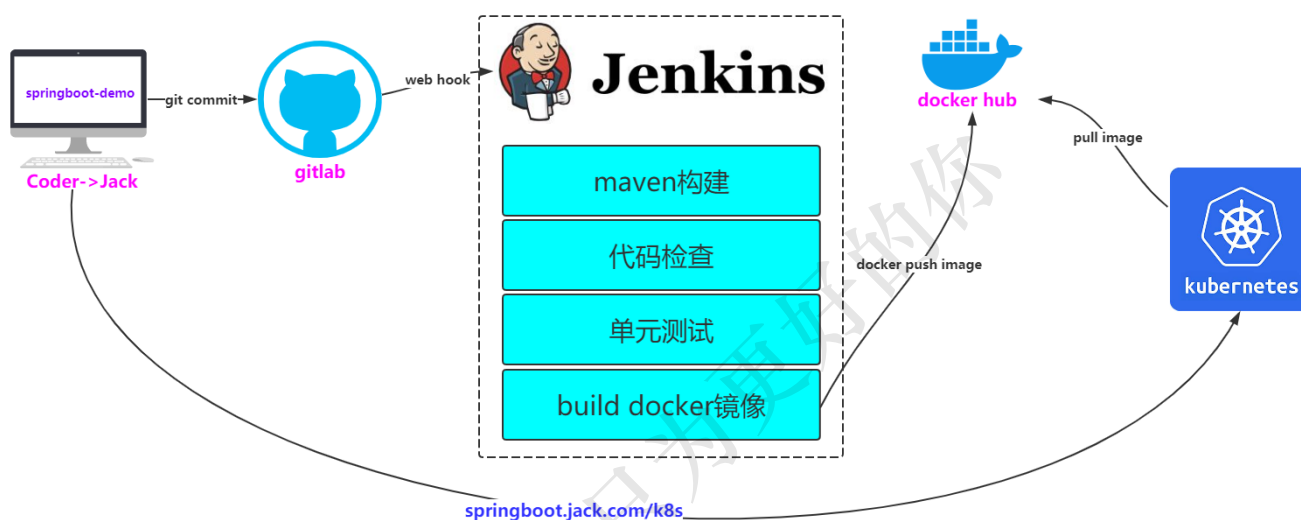


# 01 CICD

在网盘中我会提供一个md的版本[名称为“笔记命令复制伴侣.md”]

主要考虑到大家复制命令的时候，如果直接在pdf中复制，会有乱码问题，可以到md的版本中直接复制

思考：如果springboot-demo需要修改某些代码，按照上述流程，是不是又要重新打包？然后写Dockerfile，push镜像，然后写k8s配置文件等等之类的操作



思路：如果能够按照上述图解一样，在本地进行开发，然后git push到github，就能访问最终的应用该多好

## 1.1 环境准备

### 1.1.1 基础环境准备[在jenkins那台机器上安装]

- 安装java

(1)找到jdk资源上传到指定机器

```
resources/cicd/jdk-8u181-linux-x64.tar.gz
```

(2)配置环境变量

```
vim /etc/profile

export JAVA_HOME=/usr/local/java/jdk1.8.0_181
export
CLASSPATH=.:${JAVA_HOME}/jre/lib/rt.jar:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/tools
.jar
export PATH=$PATH:${JAVA_HOME}/bin

source /etc/profile

java -version
```

- 安装maven

(1)找到maven资源上传到指定机器

```
resources/cicd/apache-maven-3.6.2-bin.tar.gz
```

(2)配置环境变量

```
vim /etc/profile

export MAVEN_HOME=/usr/local/maven/apache-maven-3.6.2
export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin

source /etc/profile

mvn -version
```

(3)配置maven的阿里云镜像

```
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```

- 安装配置git

(1)下载安装

```
yum install git
```

(2)配置git

```
git config --global user.name "itcrazy2016"
git config --global user.email "itcrazy2016@163.com"
ssh-keygen -t rsa -C "itcrazy2016@163.com" --->将公钥上传到github:/root/.ssh/id_rsa.pub
```

### 1.1.1 IDEA+Spring Boot项目

#### 01 下载项目

```
git clone git@github.com:itcrazy2016/springboot-demo.git
```

#### 02 使用idea打开

此时项目已经和github关联

### 1.1.2 Gitlab

直接采用github

[git@github.com](https://github.com):itcrazy2016/springboot-demo.git

### 1.1.3 Jenkins

必须在k8s集群中，因为后面需要在jenkins的目录下创建文件执行，比如这里选用w2

#### (1)操作前须知

jenkins官网:<https://jenkins.io/>

入门指南:<<https://jenkins.io/zh/doc/pipeline/tour/getting-started/>

(1)找到对应资源: resources/cicd/jenkins.war

```
wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war
```

(2)启动jenkins[记得当前机器安装了jdk/jre，不然运行不了]

```
nohup java -jar jenkins.war --httpPort=8080 &  
tail -f nohup.out
```

(3)win浏览器访问w2的ip 121.40.56.193:8080，记录下密码，比如

```
cat /root/.jenkins/secrets/initialAdminPassword
```

(4)安装推荐的插件

# 新手入门

✓ Folders	✓ OWASP Markup Formatter	🔄 Build Timeout	🔄 Credentials Binding	<b>** Trilead API</b> <b>Folders</b> <b>** Oracle Java SE Development Kit</b> <b>Installer</b> <b>** Script Security</b> <b>** Command Agent Launcher</b> <b>OWASP Markup Formatter</b> <b>** Structs</b> <b>** Pipeline: Step API</b> <b>** Token Macro</b> <b>** bouncycastle API</b>
🔄 Timestampers	🔄 Workspace Cleanup	🔄 Ant	🔄 Gradle	
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	🔄 Pipeline: Stage View	
🔄 Git	🔄 Subversion	🔄 SSH Slaves	🔄 Matrix Authorization Strategy	
🔄 PAM Authentication	🔄 LDAP	🔄 Email Extension	🔄 Mailer	
🔄 Localization: Chinese (Simplified)				

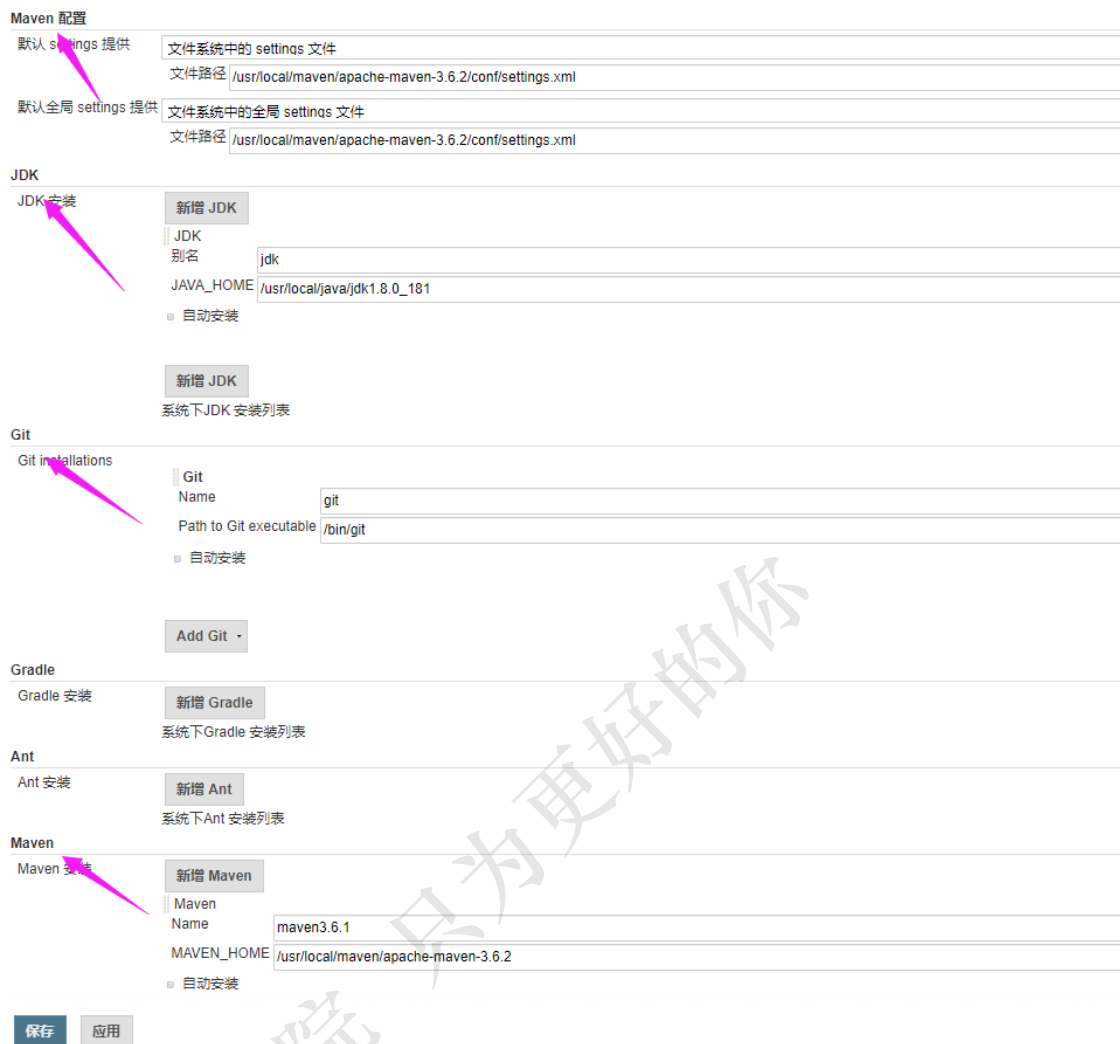
(5)创建一个用户，比如

```
username: jack
password: 123456
```

(6)安装配置git, maven

(7)在jenkins上使用centos的java, git, maven等

[系统管理]->[全局工具配置]->[Maven, JDK, Git等]



**Maven 配置**

默认 settings 提供 文件系统中的 settings 文件  
文件路径 /usr/local/maven/apache-maven-3.6.2/conf/settings.xml

默认全局 settings 提供 文件系统中的全局 settings 文件  
文件路径 /usr/local/maven/apache-maven-3.6.2/conf/settings.xml

**JDK**

JDK 安装 新增 JDK

JDK 别名 jdk  
JAVA\_HOME /usr/local/java/jdk1.8.0\_181  
☐ 自动安装

新增 JDK  
系统下 JDK 安装列表

**Git**

Git installations 新增 Git

Git Name git  
Path to Git executable /bin/git  
☐ 自动安装

Add Git

**Gradle**

Gradle 安装 新增 Gradle  
系统下 Gradle 安装列表

**Ant**

Ant 安装 新增 Ant  
系统下 Ant 安装列表

**Maven**

Maven 安装 新增 Maven

Maven Name maven3.6.1  
MAVEN\_HOME /usr/local/maven/apache-maven-3.6.2  
☐ 自动安装

保存 应用

### 1.1.4 Docker hub

使用阿里云docker镜像仓库，或者自己搭建一个  
比如使用阿里云的

```
docker login --username=itcrazy2016@163.com registry.cn-hangzhou.aliyuncs.com
```

### 1.1.5 Kubernetes集群

直接使用之前大家自己搭建的K8s集群

## 1.2 必要测试

### 1.2.1 pipeline任务

关注: /root/.jenkins/workspace目录

(1)创建jenkins的任务

### 输入一个任务名称

» 必填项



#### 构建一个自由风格的软件项目

这是Jenkins的主要功能。Jenkins将会结合任何SCM和任何构建系统来构建你的项目，甚至可以构建软件以外的系统。



#### 流水线

精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。



#### 构建一个多配置项目

适用于多配置项目，例如多环境测试，平台指定构建，等等。



#### 文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的内容，只要它们在不同的文件夹里即可。



#### GitHub 组织

扫描一个 GitHub 组织（或者个人账户）的所有仓库来匹配已定义的标记。



#### 多分支流水线

根据一个SCM仓库中检测到的分支创建一系列流水线。

确定

(2)拉取github代码，在最下面编写pipeline，然后“保存和立即构建”，同时可以查看“Console Output”

```
node {
    def mvnHome
    stage('Preparation') { // for display purposes

        git 'https://github.com/itcrazy2016/springboot-demo.git'

    }
}
```

(3)来到w2节点：ls /root/.jenkins/workspace/springboot-demo

```
[root@w2 ~]# ls /root/.jenkins/workspace/springboot-demo/
mvnw  mvnw.cmd  pom.xml  src
[root@w2 ~]#
```

(4)配置springboot-demo的task，修改pipeline内容，增加maven构建，然后“保存和立即构建”，同时可以查看“Console Output”

```
node {
    def mvnHome
    stage('Preparation') {
        git 'https://github.com/itcrazy2016/springboot-demo.git'
    }

    stage('Maven Build') {
        sh "mvn clean package"
    }
}
```

(5)来到w2节点: ls /root/.jenkins/workspace/springboot-demo

```
[root@w2 ~]# ls /root/.jenkins/workspace/springboot-demo/  
mvnw mvnw.cmd pom.xml src target  
[root@w2 ~]#
```

小结：至此，我们已经可以通过在jenkins上手动构建的方式，拿到github上的代码，并且用maven进行构建。

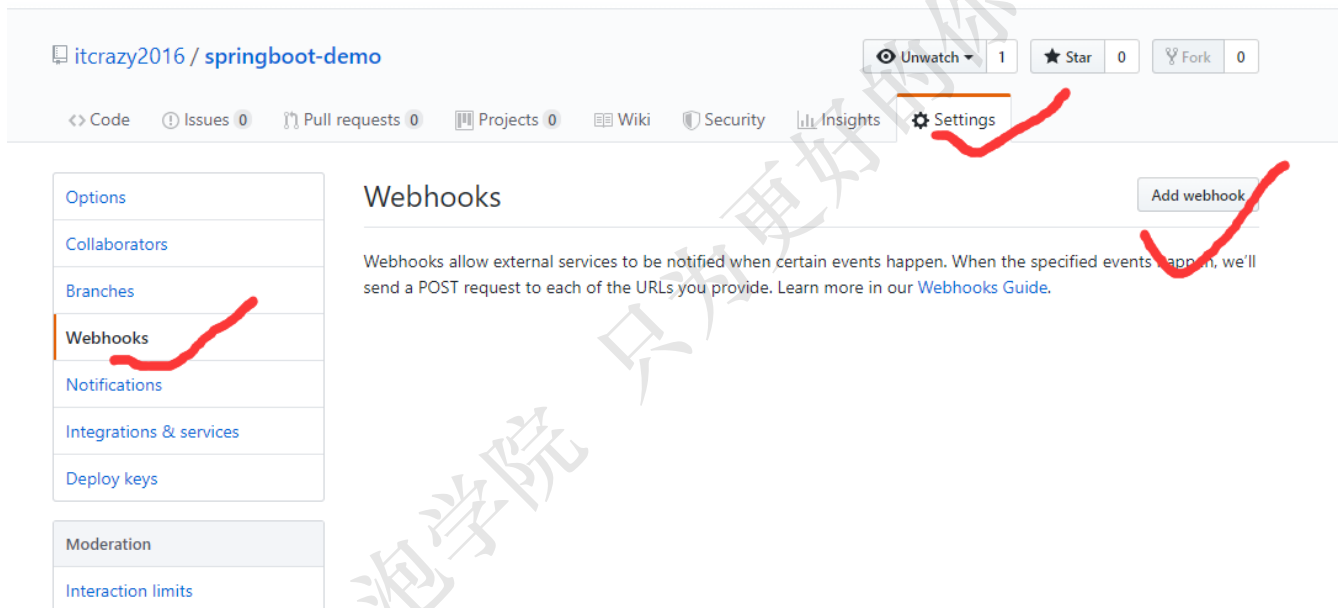
## 1.2.2 git push触发jenkins自动构建

最好的话：当用户进行git commit/push提交代码到github时，能够通知jenkins自动构建

注意：jenkins的ip一定要是github能够访问到的地址

(1)在github上配置jenkins的webhook地址

<http://121.40.56.193:8080/springboot-demo>



itcrazy2016 / springboot-demo

Unwatch1Star0Fork0

<> CodeIssues0Pull requests0ActionsProjects0WikiSecurityInsightsSettings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Secrets

Actions

Moderation

Interaction limits

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL \*

http://121.40.56.193:8080/springboot-demo

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Update webhook

Delete webhook

## (2)生成Personal access tokens

Jenkins访问github需要授权，所以在github上生成token交给jenkins使用，即 `Personal access tokens`

github的Settings[个人信息右上角]-->Developer settings-->Personal access tokens-->Generate new token

最后保存好该token，比如:72f048b514e95d6fe36f86d84374f2dcce402b43



GitHub Apps

OAuth Apps

Personal access tokens

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

What's this token for?

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> write:packages	Upload packages to github package registry
<input type="checkbox"/> read:packages	Download packages from github package registry
<input type="checkbox"/> delete:packages	Delete packages from github package registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks

### (3)jenkins安装插件

01 安装github plugin插件:[系统管理]->[插件管理]->[可选插件]

02 安装gitlab插件和gitlab hook插件:[系统管理]->[插件管理]->[可选插件]

### (4)配置GitHub Server

[系统管理]->[系统配置]->[找到github服务器]->[添加github服务器]

然后按照下面图片步骤进行操作

GitHub

GitHub 服务器

GitHub Server

名称jack

API URLhttps://api.github.com

凭据-无-添加JenkinsJenkins 凭据提供者

管理 Hook

添加 GitHub 服务器

连接测试

高级...

删除

高级...

Jenkins 凭据提供者: Jenkins

添加凭据

Domain全局凭据 (unrestricted)

类型Secret text

范围全局 (Jenkins, nodes, items, all child items, etc)

Secretgithub上生成的token

ID

描述jenkins hook token

添加取消

然后进行测试，最后点击最下面的保存

GitHub

GitHub 服务器

GitHub Server

名称jack

API URLhttps://api.github.com

凭据jenkins hook token添加

Credentials verified for user itcrazy2016, rate-limit: 4998

管理 Hook

连接测试

高级...

删除

## 1.3 核心实战走起

### 1.3.1 build&push镜像

经过前面的折腾，肯定可以获取到代码，并且用maven进行构建了，最终拿到一个target/xxx.jar

来到w2上的workspace目录：cd /root/.jenkins/workspace

(1)准备一个文件，名称为springboot-demo-build-image.sh

```
mkdir /root/.jenkins/workspace/scripts/
```

```
vi /root/.jenkins/workspace/scripts/springboot-demo-build-image.sh
```

(3)编写springboot-demo-build-image.sh文件

```
# 进入到springboot-demo目录
cd ../springboot-demo

# 编写Dockerfile文件

cat <<EOF > Dockerfile
FROM openjdk:8-jre-alpine
COPY target/springboot-demo-0.0.1-SNAPSHOT.jar /springboot-demo.jar
ENTRYPOINT ["java","-jar","/springboot-demo.jar"]
EOF

echo "Dockerfile created successfully!"

# 基于指定目录下的Dockerfile构建镜像
docker build -t registry.cn-hangzhou.aliyuncs.com/itcrazy2016/springboot-demo:v1.0 .

# push镜像, 这边需要阿里云镜像仓库登录, 在w2上登录
docker push registry.cn-hangzhou.aliyuncs.com/itcrazy2016/springboot-demo:v1.0
```

#### (4)增加pipeline

```
node {
    def mvnHome
    stage('Preparation') {
        git 'https://github.com/itcrazy2016/springboot-demo.git'
    }

    stage('Maven Build') {
        sh "mvn clean package"
    }

    stage('Build Image') {
        sh "/root/.jenkins/workspace/scripts/springboot-demo-build-image.sh"
    }
}
```

#### (5)采坑

```
# 01 文件权限
/root/.jenkins/workspace/springboot-demo@tmp/durable-7dbf7e73/script.sh: line 1:
/root/.jenkins/workspace/scripts/springboot-demo-build-image.sh: Permission denied
# 解决
chmod +x /root/.jenkins/workspace/scripts/springboot-demo-build-image.sh

# 02 docker没有运行
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon
running?
# 解决
systemctl start docker
systemctl enable docker
```

```
# 03 push权限
```

```
docker login --username=itcrazy2016@163.com registry.cn-hangzhou.aliyuncs.com
```

### 1.3.2 Kubernetes拉取镜像运行

经过前面的折腾，现在已经能够把镜像push到镜像仓库了，接下来就是k8s拉取镜像并且运行在集群中咯

根据前面的经验，肯定再创建一个stage，然后运行sh脚本，脚本中执行内容，包括yaml文件

(1)编写springboot-demo.yaml文件

在/root/.jenkins/workspace/scripts/目录下创建springboot-demo.yaml

```
# 以Deployment部署Pod
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot-demo
spec:
  selector:
    matchLabels:
      app: springboot-demo
  replicas: 1
  template:
    metadata:
      labels:
        app: springboot-demo
    spec:
      containers:
        - name: springboot-demo
          image: registry.cn-hangzhou.aliyuncs.com/itcrazy2016/springboot-demo:v1.0
          ports:
            - containerPort: 8080
---
# 创建Pod的Service
apiVersion: v1
kind: Service
metadata:
  name: springboot-demo
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: springboot-demo
---
# 创建Ingress, 定义访问规则
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: springboot-demo
```

```
spec:
  rules:
  - host: springboot.jack.com
    http:
      paths:
      - path: /
        backend:
          serviceName: springboot-demo
          servicePort: 80
```

(2)编写k8s-deploy-springboot-demo.sh文件

```
vi /root/.jenkins/workspace/scripts/k8s-deploy-springboot-demo.sh
```

```
kubectl delete -f springboot-demo.yaml

kubectl apply -f /root/.jenkins/workspace/scripts/springboot-demo.yaml

echo "k8s deploy success!"
```

(3)编写pipeline

```
node {
  def mvnHome
  stage('Preparation') {
    git 'https://github.com/itcrazy2016/springboot-demo.git'
  }

  stage('Maven Build') {
    sh "mvn clean package"
  }

  stage('Build Image') {
    sh "/root/.jenkins/workspace/scripts/springboot-demo-build-image.sh"
  }

  stage('K8S Deploy') {
    sh "/root/.jenkins/workspace/scripts/k8s-deploy-springboot-demo.sh"
  }
}
```

(4)采坑

#### # 01 权限

```
/root/.jenkins/workspace/springboot-demo@tmp/durable-8404142a/script.sh: line 1:  
/root/.jenkins/workspace/scripts/k8s-deploy-springboot-demo.sh: Permission denied
```

#### # 解决

```
chmod +x /root/.jenkins/workspace/scripts/k8s-deploy-springboot-demo.sh
```

#### # 02 worker02执行不了kubectl

切换到master上, cd ~ ---> cat .kube/config --->复制内容

切换到worker02上 cd ~ ---> vi .kube/config --->粘贴内容

(5)win的hosts文件

```
192.168.0.61 springboot.jack.com
```

咕泡学院 只为更好的你